$k-$NN and Linear Classifiers, Learning

Tomáš Svoboda and Matěj Hoffmann
thanks to Daniel Novák and Filip Železný, Ondřej Drbohlav

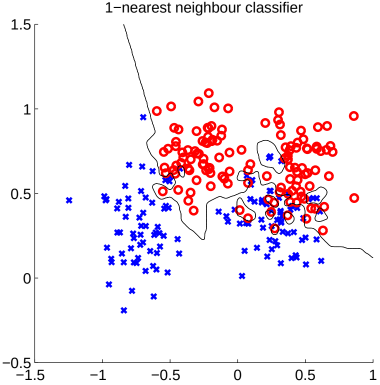
Vision for Robots and Autonomous Systems, Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague

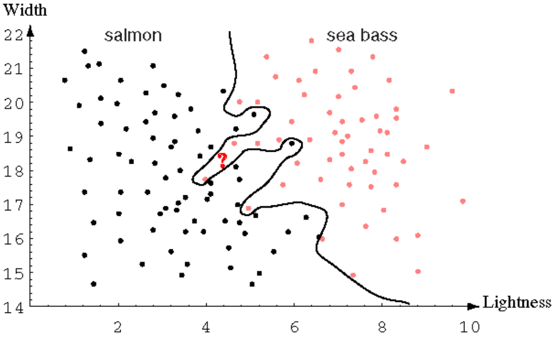
May 13, 2021

K-Nearest neighbors classification

For a query $\vec{x}$:

- Find K nearest $\vec{x}$ from the tranining (labeled) data.
- Classify to the class with the most exemplars in the set above.



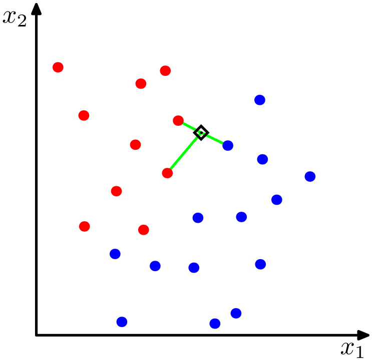


$K-$ Nearest Neighbor and Bayes $j^* = \text{argmax}_j \, P(s_j|\vec{x})$

Assume data:

- N points $\vec{x}$ in total.
- N_j points in s_j class. Hence, $\sum_j N_j = N$.

We want classify $\vec{x}$. We draw a sphere centered at $\vec{x}$ containing K points irrespective of class. V is the volume of this sphere. $P(s_j|\vec{x}) = ?$



(a)

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})}$$

$K-$ Nearest Neighbor and Bayes $j^* = \text{argmax}_j\, P(s_j|\vec{x})$

Assume data:

- N points $\vec{x}$ in total.
- N_j points in s_j class. Hence, $\sum_j N_j = N$.

We want classify $\vec{x}$. We draw a sphere centered at $\vec{x}$ containing K points irrespective of class. V is the volume of this sphere. $P(s_j|\vec{x}) =?$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})}$$

$$\begin{aligned} P(s_j) &= \frac{N_j}{N} \\ P(\vec{x}) &= \frac{K}{NV} \\ P(\vec{x}|s_j) &= \frac{K_j}{N_j V} \\ P(s_j|\vec{x}) &= \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})} = \frac{K_j}{K} \end{aligned}$$

$k - NN$ for non-parametric density estimation

$$P(\vec{x}) = \frac{K}{NV}$$

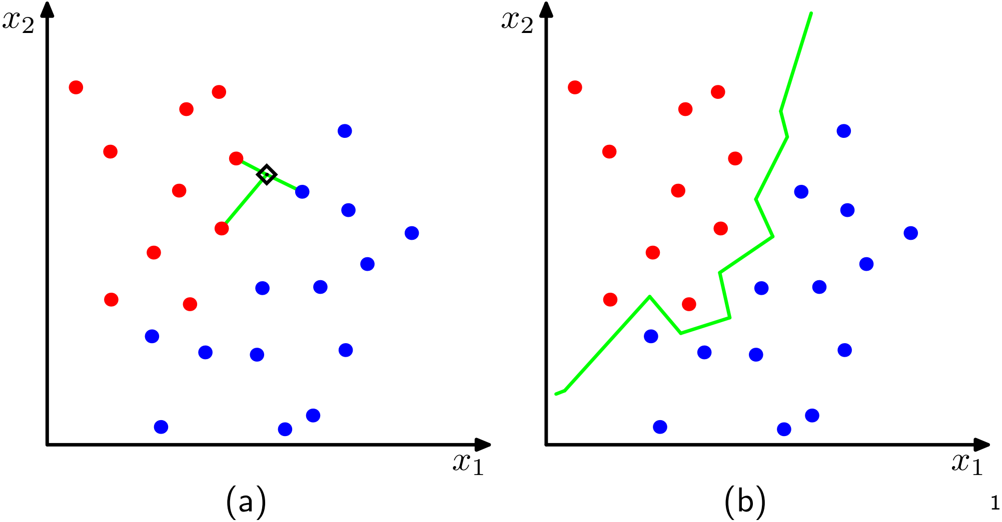
$$V = V_d R_k^d(\vec{x})$$

$R_k(\vec{x})$ - distance from $\vec{x}$ to its $k-$th nearest neighbour point (radius)

$$V_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)}$$

volume od unit $d-$dimensional sphere, Γ denotes gamma function. $V_1 = 2$, $V_2 = \pi$, $V_3 = \frac{4}{3}\pi$

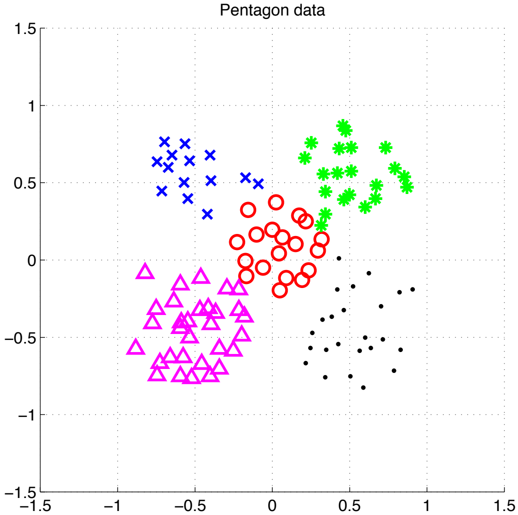
NN classification example

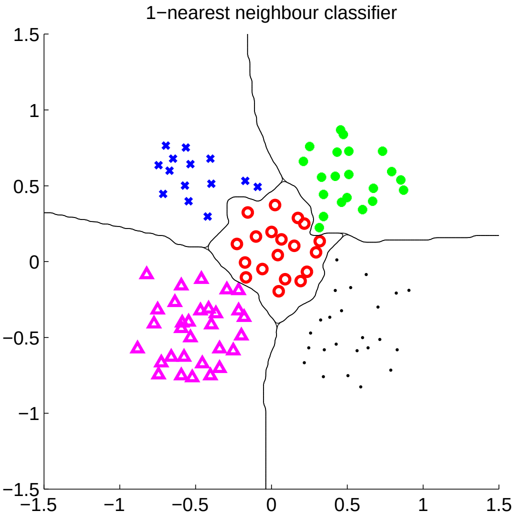


[1]

[1] Figs from [1]

NN classification example





What is *nearest*? Metrics for NN classification ...

A function D which is: nonnegative, reflexive, symmetrical, satisfying triangle inequality:

$D(\vec{a}, \vec{b}) \geq 0$

$D(\vec{a}, \vec{b}) = 0$ iff $\vec{a} = \vec{b}$

$D(\vec{a}, \vec{b}) = D(\vec{b}, \vec{a})$

$D(\vec{a}, \vec{b}) + D(\vec{b}, \vec{c}) \geq D(\vec{a}, \vec{c})$

What is *nearest*? Metrics for NN classification ...

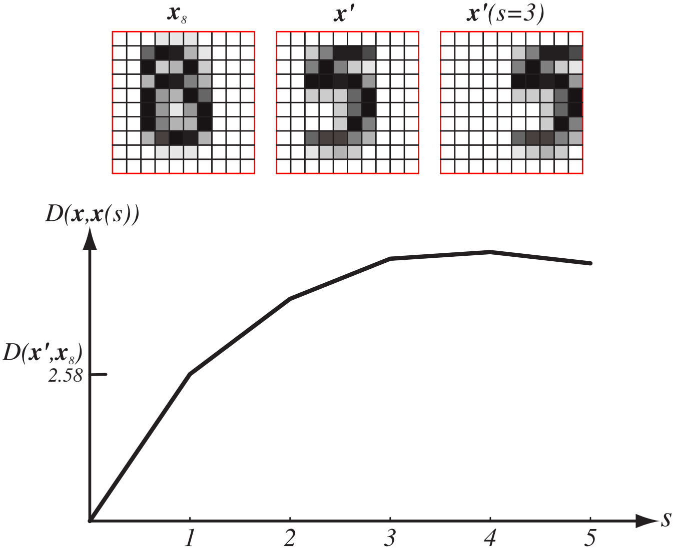
A function D which is: nonnegative, reflexive, symmetrical, satisfying triangle inequality:

$D(\vec{a}, \vec{b}) \geq 0$

$D(\vec{a}, \vec{b}) = 0$ iff $\vec{a} = \vec{b}$

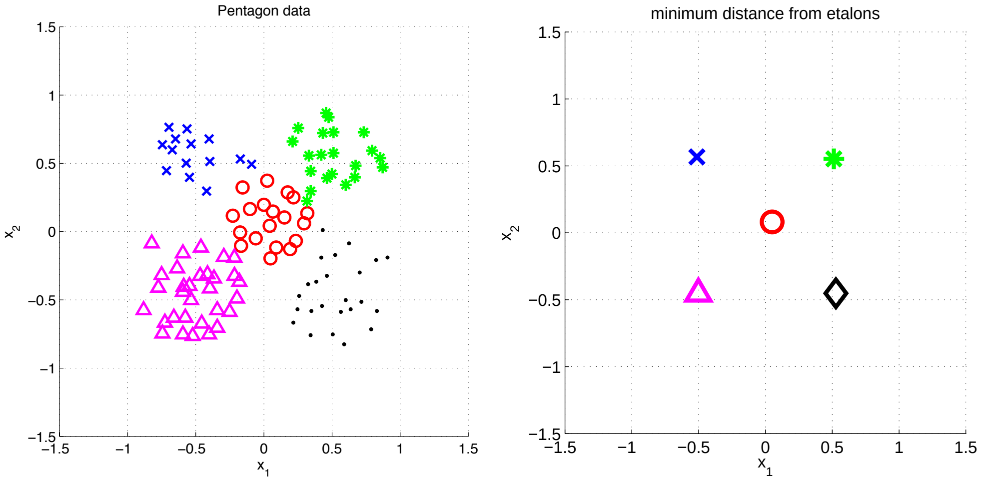
$D(\vec{a}, \vec{b}) = D(\vec{b}, \vec{a})$

$D(\vec{a}, \vec{b}) + D(\vec{b}, \vec{c}) \geq D(\vec{a}, \vec{c})$



Invariance to geometrical transformations? (figure from [3])

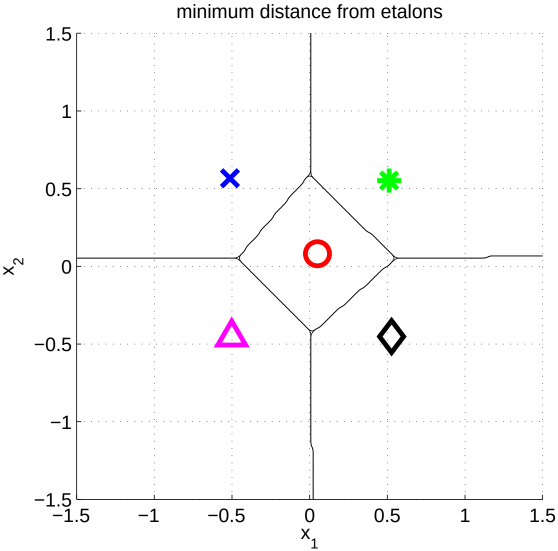
Etalon based classification



Represent $\vec{x}$ by etalon , $\vec{e}_s$ per each class $s \in S$

Separate etalons

$$s^* = \arg\min_{s \in S} \|\vec{x} - \vec{e}_s\|^2$$

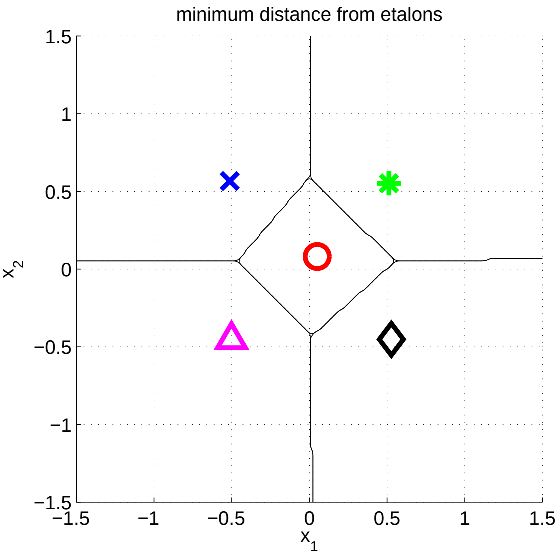


What etalons?

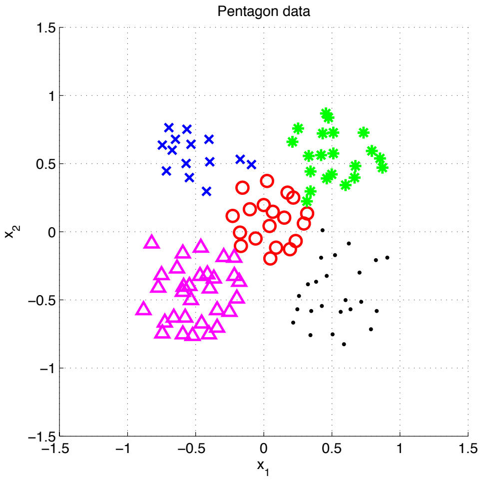
If $\mathcal{N}(\vec{x}|\vec{\mu}, \Sigma)$; all classes same covariance matrices, then

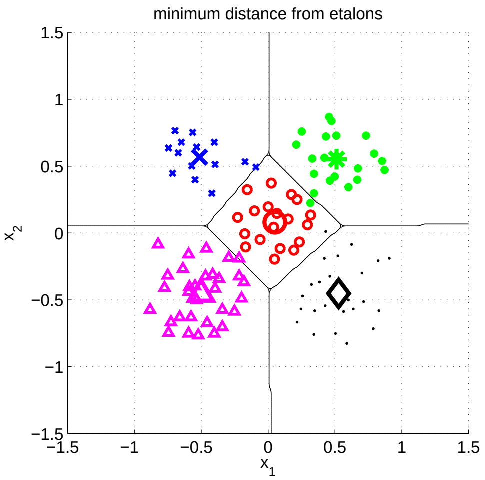
$$\vec{e}_s \stackrel{\text{def}}{=} \vec{\mu}_s = \frac{1}{|\mathcal{X}^s|} \sum_{i \in \mathcal{X}^s} \vec{x}_i^s$$

and separating hyperplanes halve distances between pairs.

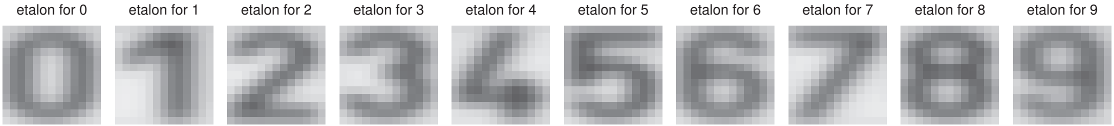


Etalon based classification, $\vec{e}_s = \vec{\mu}_s$



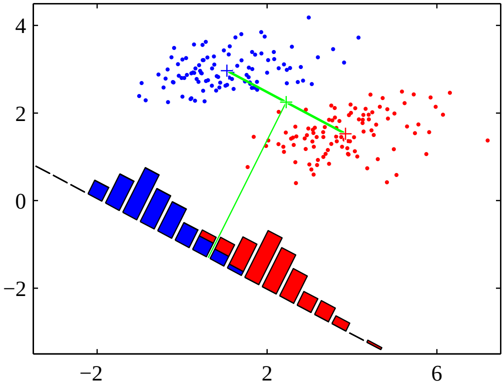


Digit recognition - etalons $\vec{e}_s = \vec{\mu}_s$



Figures from [6]

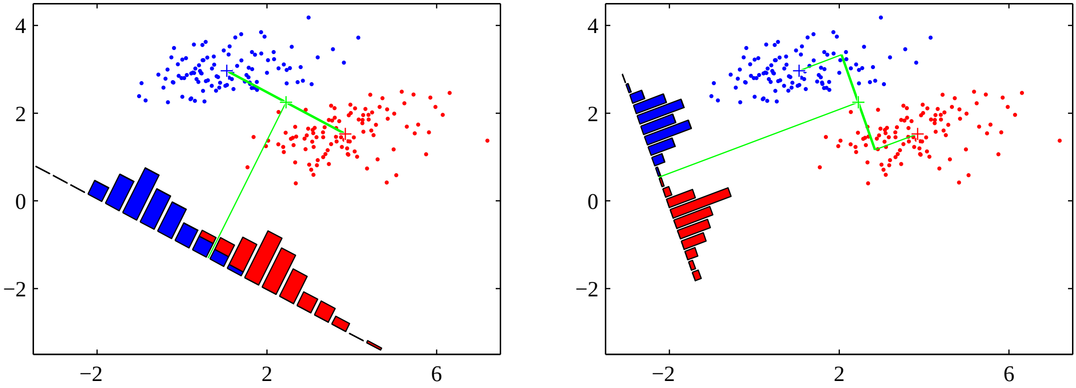
Better etalons – Fischer linear discriminant



- Dimensionality reduction
- Maximize distance between means, . . .
- . . . and minimize within class variance. (minimize overlap)

Figures from [1]

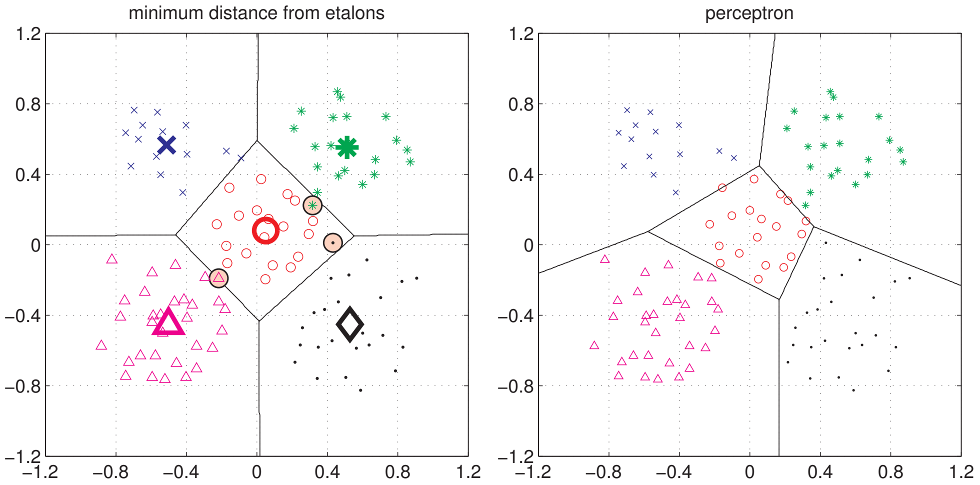
Better etalons – Fischer linear discriminant



- Dimensionality reduction
- Maximize distance between means, ...
- ...and minimize within class variance. (minimize overlap)

Figures from [1]

Better etalons?



Figures from [6]

Etalon classifier – Linear classifier

$$\begin{aligned} s^* &= \arg\min_{s\in S} \|\vec{x} - \vec{e}_s\|^2 = \arg\min_{s\in S}(\vec{x}^\top \vec{x} - 2\,\vec{e}_s^\top \vec{x} + \vec{e}_s^\top \vec{e}_s) = \\ &= \arg\min_{s\in S}\left(\vec{x}^\top \vec{x} - 2\left(\vec{e}_s^\top \vec{x} - \frac{1}{2}(\vec{e}_s^\top \vec{e}_s)\right)\right) = \\ &= \arg\min_{s\in S}\left(\vec{x}^\top \vec{x} - 2\left(\vec{e}_s^\top \vec{x} + b_s\right)\right) = \\ &= \boxed{\arg\max_{s\in S}(\vec{e}_s^\top \vec{x} + b_s)} = \arg\max_{s\in S} g_s(\vec{x}). \qquad b_s = -\frac{1}{2}\vec{e}_s^\top \vec{e}_s \end{aligned}$$

Linear function (plus offset)

$$g_s(\mathbf{x}) = \mathbf{w}_s^\top \mathbf{x} + w_{s0}$$

(1) Linear discriminant function - two class case

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

Decide s_1 if $g(\mathbf{x}) > 0$ and s_2 if $g(\mathbf{x}) < 0$

Figure from [3]

(1) Linear discriminant function - two class case

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

Decide s_1 if $g(\mathbf{x}) > 0$ and s_2 if $g(\mathbf{x}) < 0$

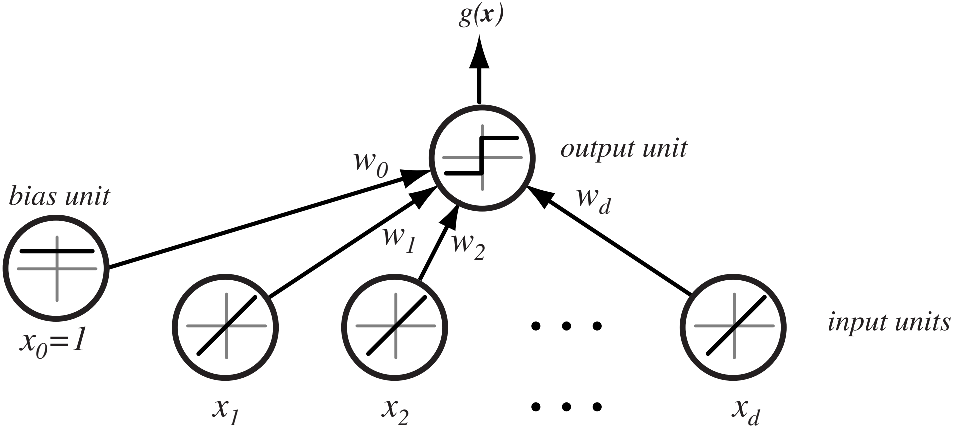


Figure from [3]

Separating hyperplane

$$\mathbf{w}^\top \mathbf{x}_1 + w_0 = \mathbf{w}^\top \mathbf{x}_2 + w_0$$

$$\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$g(\mathbf{x})$ gives an algebraic measure of the distance from $\mathbf{x}$ to the hyperplane.

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}$$

as $g(\mathbf{x}_p) = 0$,
and $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$, then:

$$g(\mathbf{x}) = r||\mathbf{w}||$$

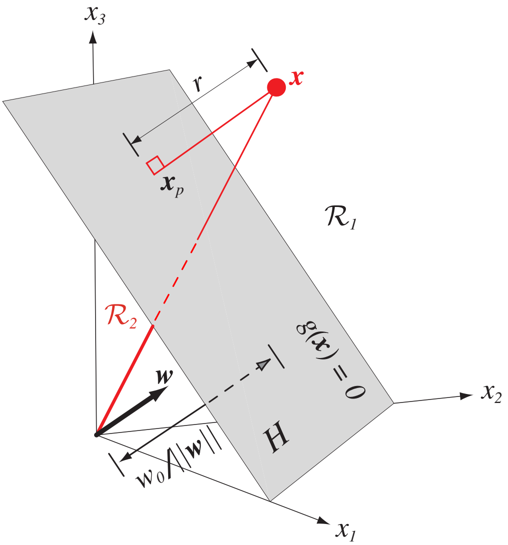


Figure from [3]

Separating hyperplane

$$\mathbf{w}^\top \mathbf{x}_1 + w_0 = \mathbf{w}^\top \mathbf{x}_2 + w_0$$

$$\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$g(\mathbf{x})$ gives an algebraic measure of the distance from $\mathbf{x}$ to the hyperplane.

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

as $g(\mathbf{x}_p) = 0$,
and $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$, then:

$$g(\mathbf{x}) = r\|\mathbf{w}\|$$

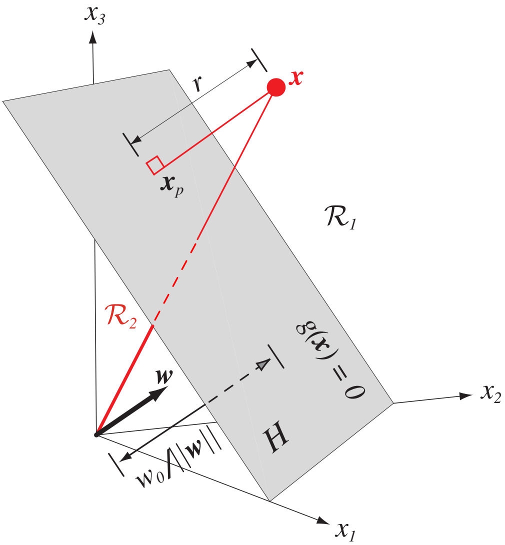


Figure from [3]

Separating hyperplane from g_1 and g_2

Etalon classifier, etalons $\vec{\mu}_1, \vec{\mu}_2$

$$
\begin{aligned}
g_1(\vec{x}) &= \vec{\mu}_1^\top \vec{x} - \frac{1}{2}\vec{\mu}_1^\top \vec{\mu}_1 \\
g_2(\vec{x}) &= \vec{\mu}_2^\top \vec{x} - \frac{1}{2}\vec{\mu}_2^\top \vec{\mu}_2
\end{aligned}
$$

Separating hyperplane:

$$g_1(\vec{x}) = g_2(\vec{x})$$

$$(\vec{\mu}_1 - \vec{\mu}_2)^\top \vec{x} = \frac{1}{2}(\vec{\mu}_1^\top \vec{\mu}_1 - \vec{\mu}_2^\top \vec{\mu}_2)$$

Two classes set-up

$|S| = 2$, i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1\,, & \text{if} \quad \mathbf{w}^\top \mathbf{x} + w_0 > 0\,, \\ s = -1\,, & \text{if} \quad \mathbf{w}^\top \mathbf{x} + w_0 < 0\,. \end{cases}$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

Two classes set-up

$|S| = 2$, i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1\,, & \text{if} \quad \mathbf{w}^\top \mathbf{x} + w_0 > 0\,, \\ s = -1\,, & \text{if} \quad \mathbf{w}^\top \mathbf{x} + w_0 < 0\,. \end{cases}$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \ \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

Two classes set-up

$|S| = 2$, i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1\,, & \text{if} \quad \mathbf{w}^\top \mathbf{x} + w_0 > 0\,, \\ s = -1\,, & \text{if} \quad \mathbf{w}^\top \mathbf{x} + w_0 < 0\,. \end{cases}$$

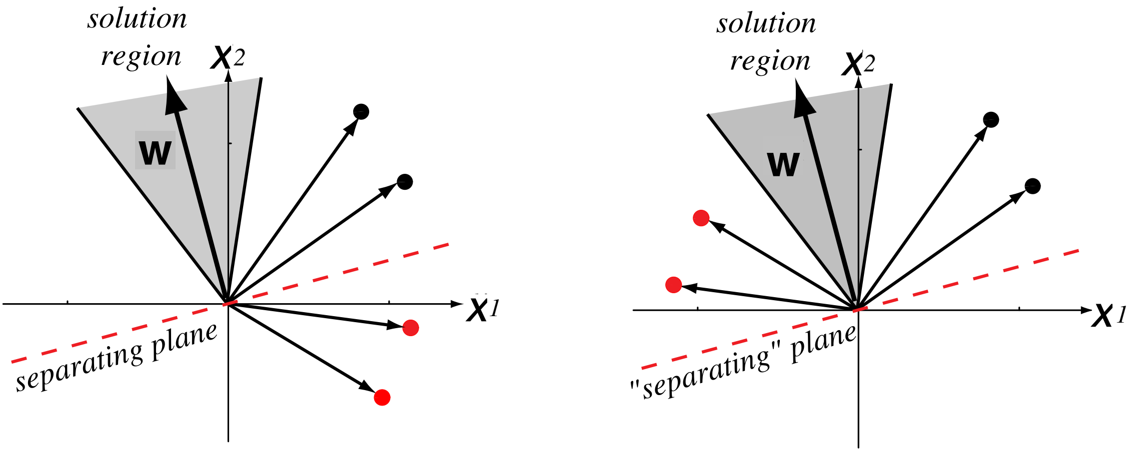
$$\boxed{\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix},\ \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}}$$

for all $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

Solution (graphically)



Four training samples. Left: orginal, Right: sign corrected

Figure from [3] (notation changed)

Learning $\mathbf{w}$, gradient descent

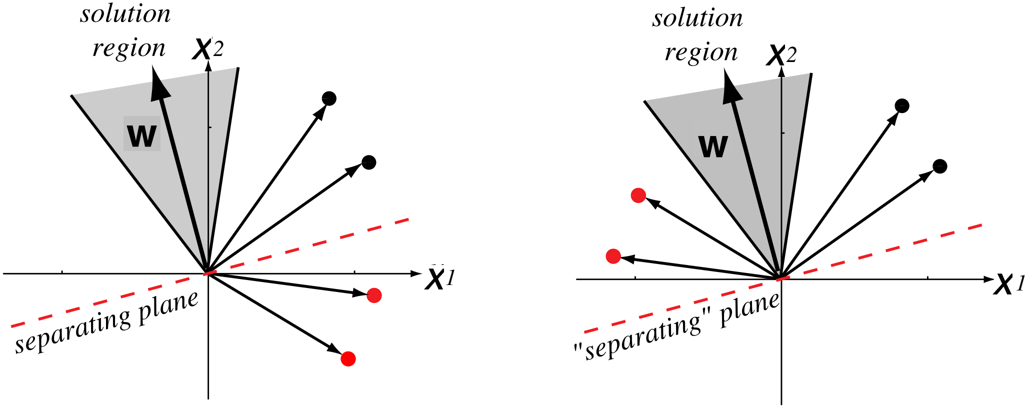
A criterion to be minimized $J(\mathbf{w})$; assume to be known

Initialize $\mathbf{w}$, threshold θ, learning rate α
$k \leftarrow 0$
repeat
 $k \leftarrow k + 1$
 $\mathbf{w} \leftarrow \mathbf{w} - \alpha(k)\nabla J(\mathbf{w})$
until $|\alpha(k)\nabla J(\mathbf{w})| < \theta$
return $\mathbf{w}$

Learning **w** - Perceptron criterion

Goal: Find a weight vector $\mathbf{w} \in \Re^{D+1}$ (original feature space dimensionality is D) such that:

$$\mathbf{w}^\top \mathbf{x}_j > 0 \qquad (\forall j \in \{1, 2, ..., m\})$$



(Perceptron) Criterion to be minimized:

Learning **w** - Perceptron criterion

Goal: Find a weight vector $\mathbf{w} \in \Re^{D+1}$ (original feature space dimensionality is D) such that:

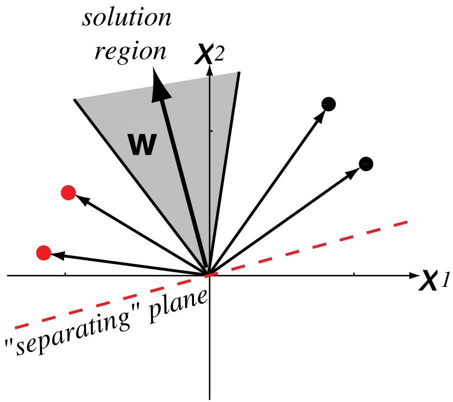
$$\mathbf{w}^\top \mathbf{x}_j > 0 \qquad (\forall j \in \{1, 2, ..., m\})$$

(Perceptron) Criterion to be minimized:

$$J(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{w}^\top \mathbf{x}$$

where $\mathcal{X}$ is a set of missclassified $\mathbf{x}$.

$$\nabla J(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{x}$$



(Batch) Perceptron algorithm

Initialize $\mathbf{w}$, threshold θ, learning rate α
$k \leftarrow 0$
repeat
    $k \leftarrow k + 1$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha(k) \sum_{\mathbf{x} \in \mathcal{X}(k)} \mathbf{x}$
until $|\alpha(k) \sum_{\mathbf{x} \in \mathcal{X}(k)} \mathbf{x}| < \theta$
return $\mathbf{w}$

Fixed-increment single-sample Perceptron

n patterns/samples, we are looping over all patterns repeatedly

Initialize $\mathbf{w}$

$k \leftarrow 0$

repeat

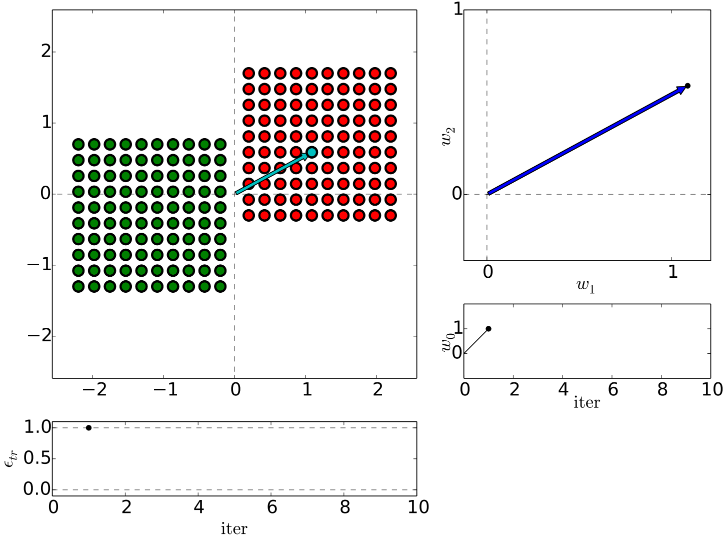
$k \leftarrow (k+1) \bmod n$

if $\mathbf{x}^k$ missclassified, **then** $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

until all $\mathbf{x}$ correctly classified

return $\mathbf{w}$

Perceptron iterations/loops



n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$

$k \leftarrow 0$

repeat

$k \leftarrow (k+1) \bmod n$

if $\mathbf{x}^k$ missclassified, **then**

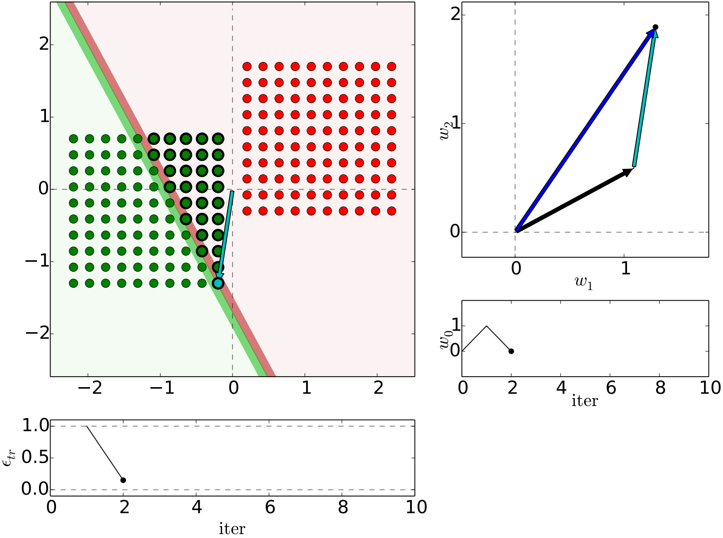
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

until all $\mathbf{x}$ correctly classified

return $\mathbf{w}$

(Dark) Blue is **w** after update step. Reds are $+$, Greens $-$.

Perceptron iterations/loops



n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$

$k \leftarrow 0$

repeat

$k \leftarrow (k+1) \bmod n$

if $\mathbf{x}^k$ missclassified, **then**

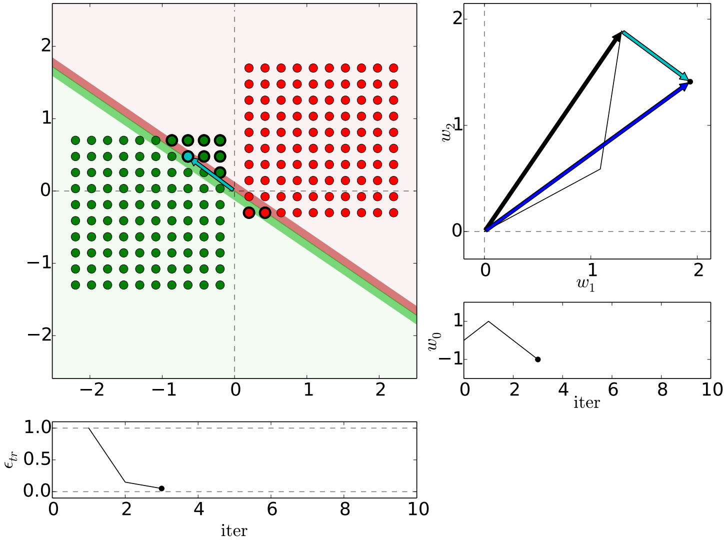
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

until all $\mathbf{x}$ correctly classified

return $\mathbf{w}$

(Dark) Blue is $\mathbf{w}$ after update step. Reds are $+$, Greens $-$.

Perceptron iterations/loops



n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$

$k \leftarrow 0$

repeat

$k \leftarrow (k+1) \bmod n$

if $\mathbf{x}^k$ missclassified, **then**

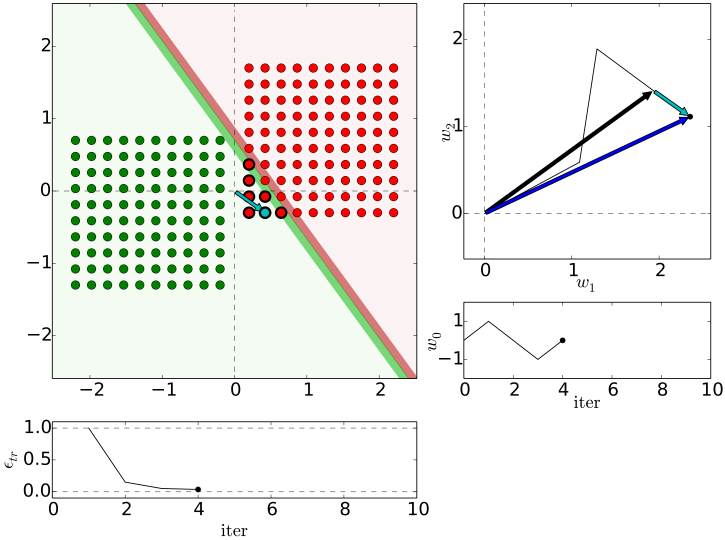
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

until all $\mathbf{x}$ correctly classified

return $\mathbf{w}$

(Dark) Blue is $\mathbf{w}$ after update step. Reds are $+$, Greens $-$.

Perceptron iterations/loops



n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$

$k \leftarrow 0$

repeat

$k \leftarrow (k+1) \bmod n$

if $\mathbf{x}^k$ missclassified, **then**

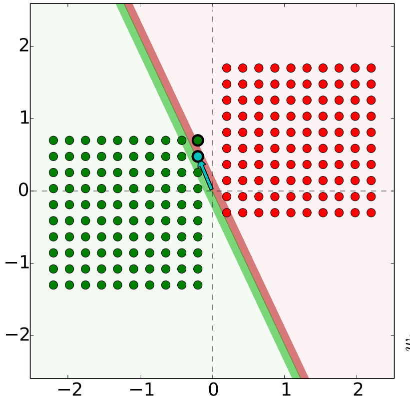
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

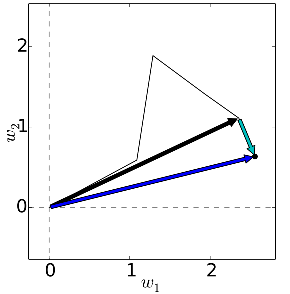
until all $\mathbf{x}$ correctly classified

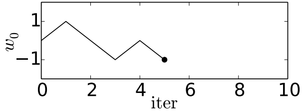
return $\mathbf{w}$

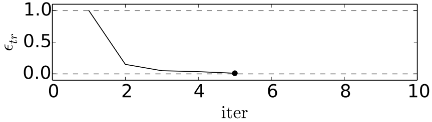
(Dark) Blue is $\mathbf{w}$ after update step. Reds are $+$, Greens $-$.

Perceptron iterations/loops









n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$

$k \leftarrow 0$

repeat

$k \leftarrow (k+1) \bmod n$

if $\mathbf{x}^k$ missclassified, **then**

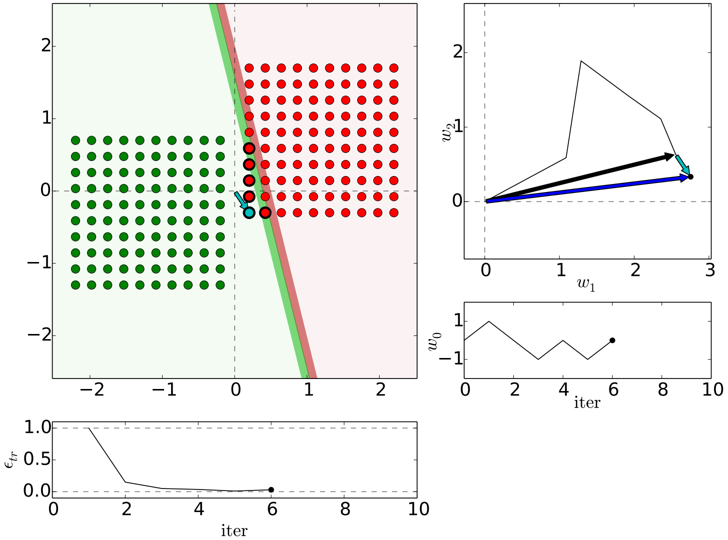
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

until all $\mathbf{x}$ correctly classified

return $\mathbf{w}$

(Dark) Blue is $\mathbf{w}$ after update step. Reds are $+$, Greens $-$.

Perceptron iterations/loops

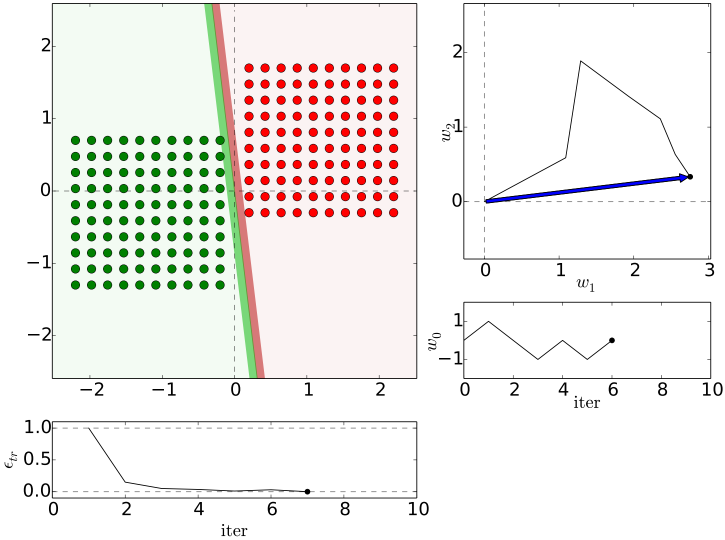


n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$
$k \leftarrow 0$
repeat
$k \leftarrow (k+1) \bmod n$
if $\mathbf{x}^k$ missclassified, **then**
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$
until all $\mathbf{x}$ correctly classified
return $\mathbf{w}$

(Dark) Blue is $\mathbf{w}$ after update step. Reds are $+$, Greens $-$.

Perceptron iterations/loops

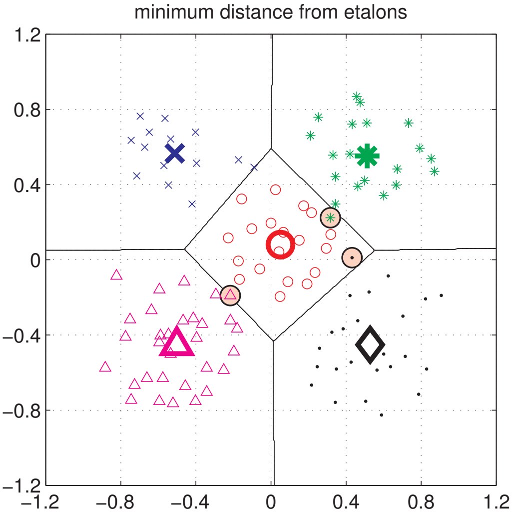


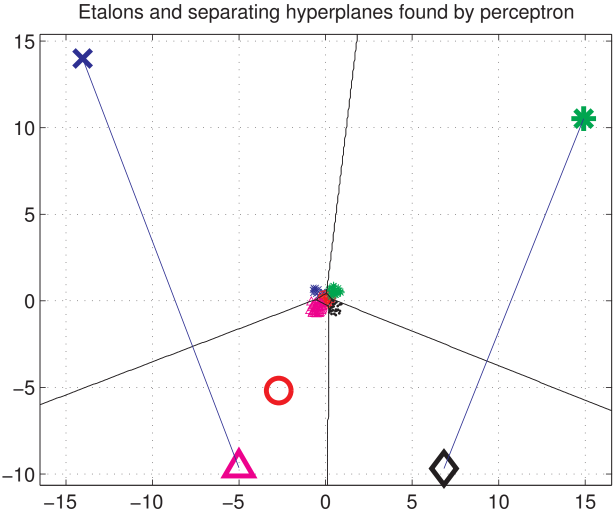
n patterns/samples, we are looping over all patterns repeatedly:

Initialize $\mathbf{w}$
$k \leftarrow 0$
repeat
  $k \leftarrow (k+1) \bmod n$
  if $\mathbf{x}^k$ missclassified, **then**
    $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$
until all $\mathbf{x}$ correctly classified
return $\mathbf{w}$

(Dark) Blue is $\mathbf{w}$ after update step. Reds are $+$, Greens $-$.

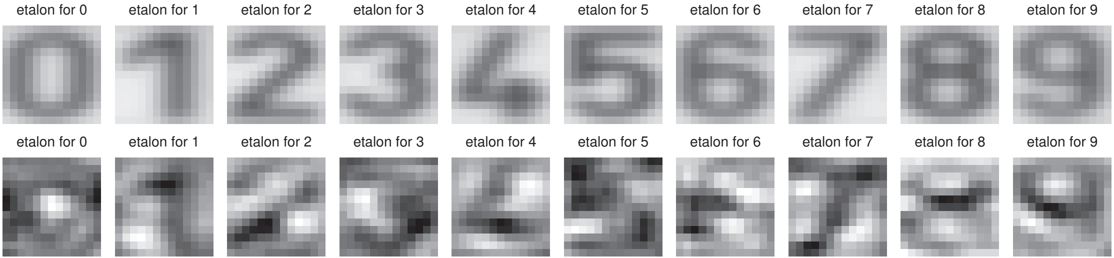
Etalons: means vs. found by perceptron





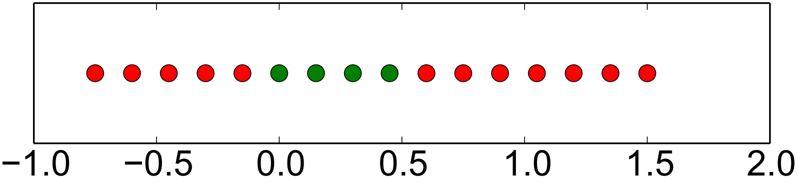
Figures from [6]

Digit recognition - etalons means vs. perceptron



Figures from [6]

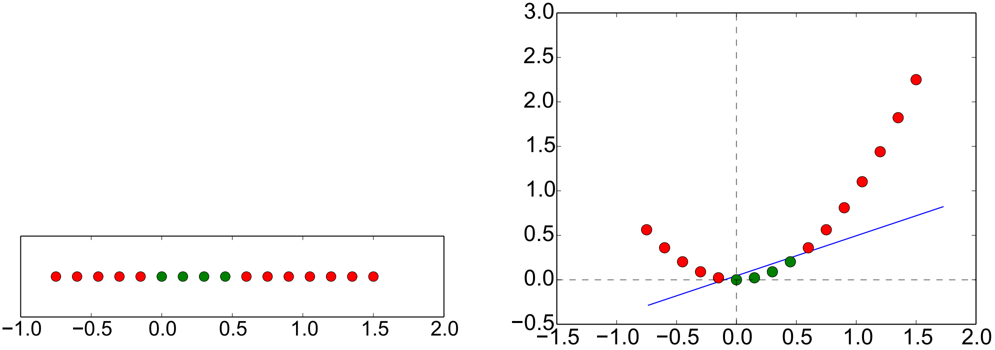
What if not lin separable?



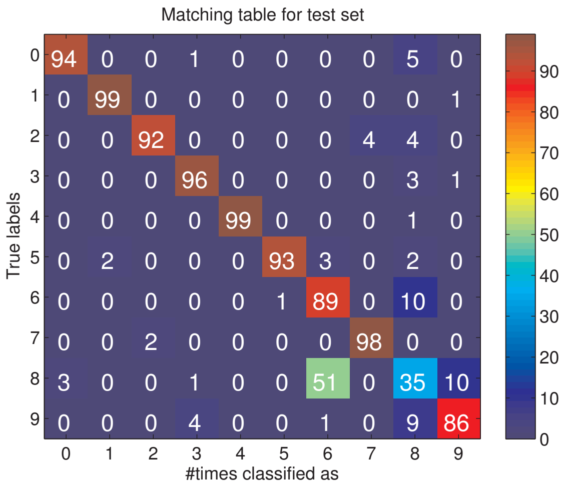
Dimension lifting

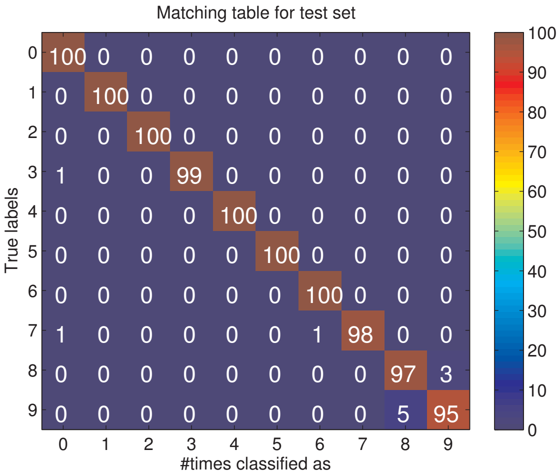
$$\mathbf{x} = [x, x^2]^\top$$

Dimension lifting, $\mathbf{x} = [x, x^2]^\top$



Performance comparison, parameters fixed





Learning and decision

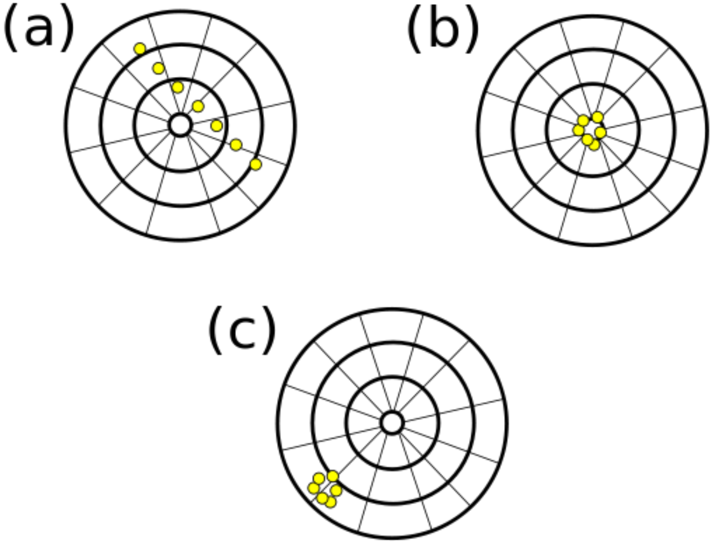
Learning stage - learning models/function/parameters from data.

Decision stage - decide about a query $\vec{x}$.

What to learn?

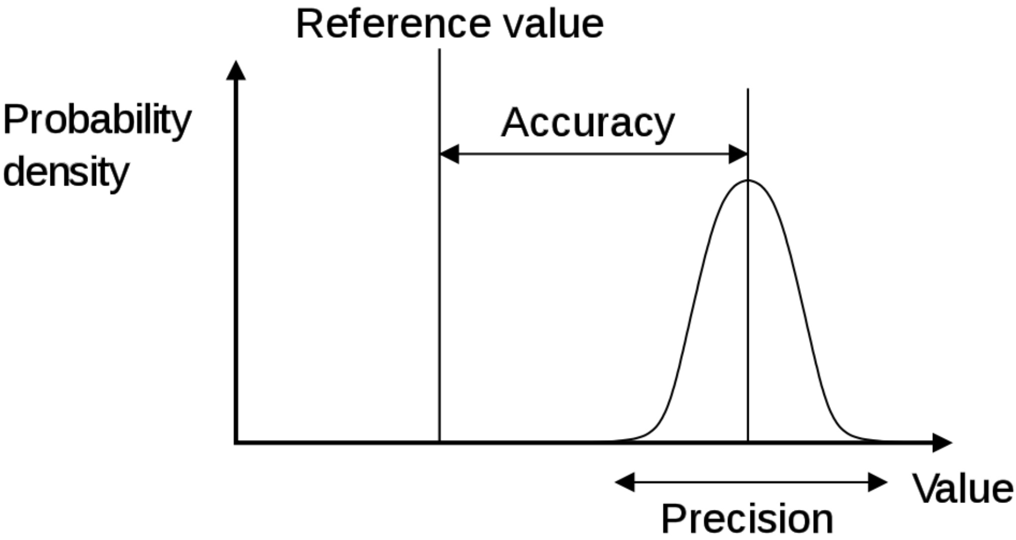
- Generative model : Learn $P(\vec{x}, s)$. Decide by computing $P(s|\vec{x})$.
- Discriminative model : Learn $P(s|\vec{x})$
- Discriminant function : Learn $g(\vec{x})$ which maps $\vec{x}$ directly into class labels.

Accuracy vs precision



https://commons.wikimedia.org/wiki/File:Precision_versus_accuracy.svg

Accuracy vs precision



https://en.wikipedia.org/wiki/Accuracy_and_precision

References I

Further reading: Chapter 18 of [5], or chapter 4 of [1], or chapter 5 of [3]. Many figures created with the help of [4]. You may also play with demo functions from [6].

[1] Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer Science+Bussiness Media, New York, NY, 2006.
https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

[2] Yen-Chi Chen.
Lecture 7: Density estimation: k-nearest neighbor and basis approach.
Lecture within STAT 425: Introduction to Nonparametric Statistics, 2018.
http://faculty.washington.edu/yenchic/18W_425/Lec7_knn_basis.pdf.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork.
Pattern Classification.
John Wiley & Sons, 2nd edition, 2001.

References II

[4] Votjěch Franc and Václav Hlaváč.
Statistical pattern recognition toolbox.
http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html.

[5] Stuart Russell and Peter Norvig.
Artificial Intelligence: A Modern Approach.
Prentice Hall, 3rd edition, 2010.
http://aima.cs.berkeley.edu/.

[6] Tomáš Svoboda, Jan Kybic, and Hlaváč Václav.
Image Processing, Analysis and Machine Vision — A MATLAB Companion.
Thomson, Toronto, Canada, 1st edition, September 2007.
http://visionbook.felk.cvut.cz/.